On the Computational Power of Binary Decision Diagram with Redundant Variables

AUTHOR(S):

YAMADA, Tetsuya; YASUURA, Hiroto

CITATION:

YAMADA, Tetsuya ...[et al]. On the Computational Power of Binary Decision Diagram with Redundant Variables. 数理解析研究所講究録 1994, 871: 80-86

ISSUE DATE:

1994-05

URL:

http://hdl.handle.net/2433/84053

数理解析研究所講究録
第871巻 1994年 80-86

On the Computational Power of Binary Decision Diagram with Redundant Variables

Tetsuya YAMADA (山田 哲也) Hiroto YASUURA (安浦 寛人)

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University

Abstract

We introduce a BDD with redundant variables as an Indexed BDD(IBDD) and define PolyIBDD as the class of Boolean functions represented by polynomial-sized IBDDs of the number of input variables. Assuming that the class of languages on $\{0,1\}^*$ that are accepted by logarithmic space bounded DTMs is DLOG, the following relation holds. $PolyIBDD = DLOG$. That is to say that languages which belong to DLOG also belong to PolyIBDD.

Keywords: Binary Decision Diagram, Indexed BDD, $PolyI^{k(n)}BDD$, PolyIBDD, DLOG

1 Introduction

Due to the recent progress of VLSI technology, it is desired to design large scaled and complicated logic circuits. Computer-aided design for digital circuits (CAD) is indispensable to design such a large scaled integrated digital circuits. Representations of Boolean functions affect area and computational time in synthesis and simulation of the logic circuit. The importance of representations is increasing as the scale of Boolean functions is becoming larger.

Binary Decision Diagram(BDD) is one of the data structure to represent Boolean functions[1, 2]. A BDD is a directed acyclic graph which is constructed from successive Shannon expansion[1] of a Boolean function according to a given input variable order. When isomorphic subgraphs and redundant nodes are reduced, a Boolean function has a canonical form of BDDs given an input variable order. That is two functions are equivalent if and only if the reduced BDD for each function is isomorphic given an input variable order. The canonicity makes easy to check equivalence of Boolean functions. Besides many practical Boolean functions are representable in feasible memory space using BDDs. Due to good properties above, BDD is widely used in CAD, such as logic synthesis, test generation and verification.

The number of nodes of BDDs (it is also called "size ") affects the processing speed of Boolean functions and memory area. Thus we can use BDD's size as complexity measure of Boolean functions. The largest size of BDDs is at most polynomial to the number of input variables to manipulate Boolean functions.

The term "Sequence of BDD" can be defined as an infinite set of BDDs which are characterized by a size parameter i. A sequence of BDDs is an infinite set of BDDs such that each BDD represents a Boolean function of i input variables for each positive number i. We will regard an on-set of a Boolean function which is represented by a BDD in a sequence of BDDs as a string on $\{0,1\}^*$. Now there is one-to-one correspondence between a sequence of BDD and a formal language on $\{0,1\}^*$, thus we can argue the sequence of BDDs using results of formal language theory.

Ishiura et al.[4] defined the class of Boolean functions which are represented by a sequence of polynomial-sized BDDs of the number of input variables as PolyBDD. They have showed relations between PolyBDD and Turing machines as follows: $Regular\ Set \subseteq PolyBDD \subseteq DLOG$, where $DLOG$ is the class of languages that are accepted by logarithmic space-bounded deterministic Turing machines.

Jain et al.[5] proposed an Indexed BDD(IBDD) as a more compact and generalized form of BDDs. An IBDD is a BDD with redundant input variables. Although IBDD has no canonical form for any Boolean function and a construction of IBDD is a complicated task, an integer multiplier and an HWB(Hidden Weighted Bit) function [3] can be represented by polynomial-sized IBDDs of the number of input variables. It is known that BDD requires at least exponential to represent the same functions[3].

We will define the class of Boolean functions which are represented by a sequence of polynomial-sized IBDDs of the number of input variables as PolyIBDD. The purpose of this paper is to clarify the relation between PolyIBDD and well-known classes of formal languages.

In this paper, we present relations between IBDDs and Turing machines. The following relation holds.

$$PolyIBDD = DLOG$$

In Section 2, we define a sequence of IBDDs, PolyIBDD, and these related terms. In Section 3, we show that PolyIBDD=DLOG. In Section 4, we give concluding remarks.

[1] Shannon expansion of variable x_i means $F = \overline{x_i} \cdot F_{x_i=0} + x_i \cdot F_{x_i=1}$.

2 Definitions

2.1 BDD and Indexed BDD

Definition 1 (Indexed BDD:IBDD)
An IBDD is an 8-tuple $IB = (N_v, N_c, init, index, k(n), e_0, e_1, level)$
given variable sets $\pi = \{x_1, x_2, \ldots, x_n\}$, where

- N_v : a set of variable nodes
- N_c : a set of constant nodes
- $N_v \cup N_c$: a set of nodes
- $init \in N_v$: an initial node
- $index : (N_v \cup N_c) \rightarrow (\pi \cup \{0,1\})$ the index of a node
  - $index(v) \in \pi, \quad if\ v \in N_v$
  - $index(v) \in \{0,1\}, \quad if\ v \in N_c$
- $k(n)$: the number of layers
- $e_0(v) : N_v \rightarrow (N_v \cup N_c)$ the '0' edge directed from a node v
- $e_1(v) : N_v \rightarrow (N_v \cup N_c)$ the '1' edge directed from a node v
- A level expressing a tuple (L, l).

  L : a layer number$(1 \leq L \leq k(n)+1)$,

  l : a level in a layer $(1 \leq l \leq |\pi|)$
  - $\varphi : (L \times l) \rightarrow \pi \cup \{0,1\}$
  - $\varphi_i : l \rightarrow \pi \cup \{0,1\}$
  - $\varphi(i,j) = \varphi_i(j)$
  - $index(init) = \varphi(1,1) = \varphi_1(1)$
  - $\varphi_i^{-1}(index(v)) = \varphi_i^{-1}(index(v'))$, $iff\ index(v) = index(v')$, where $v, v' \in N_v$
  - $index(v) = \varphi(k(n)+1, 1) = \varphi_{k(n)+1}(1)$, $iff\ v \in N_c$
  - $e_0(v)$ satisfies any of the followings.
    $index(v) = \varphi(i,j), \quad if\ v \in N_v$
    $1 \leq i \leq k(n), 1 \leq j \leq |\pi|$
    1. $index(e_0(v)) = \varphi(i,j') \quad 1 \leq j < j' \leq |\pi|$
    2. $index(e_0(v)) = \varphi(i',j'') \quad 1 \leq i < i' \leq k(n)$
    3. $index(e_0(v)) = \varphi(k(n)+1, 1)$

    $e_1(v)$ is defined similarly.

Each node $v, (v \in N_v \cup N_c)$ represents a Boolean function $f_v : \{0,1\}^{|\pi|} \rightarrow \{0,1\}$.

$$f_v = \begin{cases} 0 & if\ \ v \in N_c,\ index(v) = 0 \\ 1 & if\ \ v \in N_c,\ index(v) = 1 \\ \overline{index(v)} \cdot f_{e_0}(v) + & \\ index(v) \cdot f_{e_1}(v) & if\ \ v \in N_v \end{cases}$$

An IBDD IB represents a Boolean function f_B as $f_B = f_{init}$, where $x_i^1 = x_i^2 = \cdots = x_i^j$ $(1 \leq i \leq |\pi|, 1 \leq j \leq k(n))$ □

IBDD differs from BDD in the following points.

- A variable order is free on each layer.
- Variables at different level on a layer are different.
- Different graphs which represent the same Boolean function are obtained if layer number $k(n)$ varies. i.e. there is no canonical form.
- An IBDD equals a BDD when a layer number k=1.

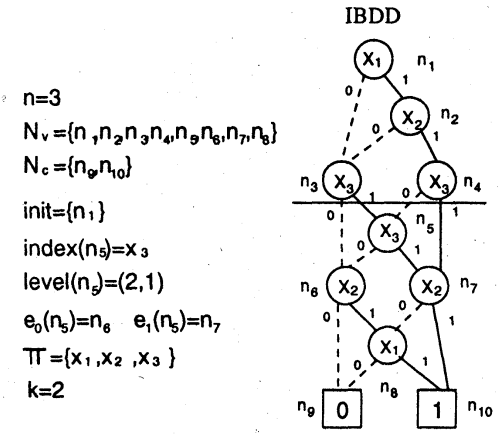

Figure 1: IBDD IB_3^2

We define a sequence of IBDDs following the definition of a sequence of Boolean functions[2] in [7]. A sequence of IBDDs is an infinite set of IBDDs which are characterized by a size parameter n.

Definition 2 (Sequence of IBDDs)
A sequence of IBDDs $\{IB_n^{k(n)}\}$ is an infinite set of IBDDs $IB_1^{k(1)}$, $IB_2^{k(2)}$, ..., where $IB_n^{k(n)}$ is an IBDD which represents a Boolean function of n variables. $k(i)$ is a function which is related to i. □

We define a formal language $L_{\{IB_n^{k(n)}\}}$ for a sequence of IBDDs $\{IB_n^{k(n)}\}$ following the definition of a formal language for a sequence of Boolean functions in [7].

Definition 3 $(L_{\{IB_n^{k(n)}\}})$
A formal language $L_{\{IB_n^{k(n)}\}}$ for a sequence of IBDDs is

$$\{L_{\{IB_n^{k(n)}\}} = f^{-1}_{IB_n^{k(n)}}(1) | n = 1, 2, \ldots\},$$

where $f_{IB_n^{k(n)}}$ denotes a Boolean function represented by an IBDD $IB_n^{k(n)}$, and $f^{-1}_{IB_n^{k(n)}}(1)$ means the set of n-length sequence $\{b_1, \ldots, b_n\} \in \{0,1\}^n$ which satisfies $f_{IB_n^{k(n)}}(b_1, b_2, \ldots, b_n) = 1$. □

We can connect IBDDs with formal languages from the definition above. However, we need to define uniformity of a sequence of IBDDs as a family of logic

[2] A sequence of Boolean functions is often called a family of Boolean functions.

circuits as defined in [6] in order to discuss the relation between IBDDs and Turing machines. We define an encoding of an IBDD according to an encoding of logic circuits[6]. First of all, we begin with a description of each node of an IBDD D_v.

Definition 4 (node description of IBDD D_v)
Each node $v, (v \in N_v \cup N_c)$ of an IBDD is described a 4-tuple $D_v = (index(v), level, e_0(v), e_1(v))$. We can describe an entire IBDD when we gather descriptions of all nodes of an IBDD. We call the entire BDD $D_B(D_{IB})$. □

Definition 5 (standard encoding of IBDD)
Let C be a constant, and $size(IB_n^k)$ size of an IBDD $IB_n^{k(n)}$. Assuming that each node description of an IBDD D_v is encoded in a sequence of $\{0,1\}^*$ with $Clog\ (size(IB_n^{k(n)}))$ length and encoded descriptions of D_v differs from each other, we define this encoding as *standard encoding* of the IBDD $IB_n^{k(n)}$ and denote $\overline{IB_n^{k(n)}}$. □

A Node description D_v of a polynomial-sized IBDD $IB_n^{k(n)}$ consists of $index, level(L,l)$, $e_0(v)$ and $e_1(v)$. The number of them are n, $poly(n)$[3], $poly(n)$, and $poly(n)$ respectively. Then D_v is encoded in $O(log\ n)$ space. We can define a uniformity of a sequence of IBDDs $\{IB_n^{k(n)}\}$.

Definition 6 (uniformity of IBDD)
Let $\overline{IB_n^{k(n)}}$ be a standard encoding of n-th IBDD $IB_n^{k(n)}$ of a sequence of IBDDs $\{IB_n^{k(n)}\}$.

Assuming that the function which generates $\overline{IB_n^{k(n)}}$ from a binary description of n is computable with a logarithmic space bounded deterministic Turing machine, we define a sequence of IBDDs $\{IB_n^{k(n)}\}$ as *logarithmic uniform*. □

Definition 7 ($PolyI^{k(n)}BDD$)
Assuming that each IBDD $IB_i^{k(i)}$ of a sequence of IBDDs $\{IB_n^{k(n)}\}$ satisfies $size(IB_i^{k(i)}) \leq poly(i)$, a logarithmic uniform sequence of IBDD $\{IB_n^{k(n)}\}$ shall be denoted as $PolyI^{k(n)}BDD$. □

Definition 8 ($PolyIBDD$)
PolyIBDD is the class of languages that represented by IBDDs and each IBDD's size is polynomial to the number of input variables: i.e.

$$\bigcup_{k(n)=n^{O(1)}} L_{PolyI^{k(n)}BDD} = PolyIBDD.$$

□

[3] $poly(n)$ is a function of $n^{O(1)}$, i.e. $poly(n)$ follows a polynomial growth of n.

2.2 DLOG and Logarithmic Space Bounded Deterministic Turing Machine

We define M as a deterministic Turing machine with a two-way read-only input tape, two-way log-space bounded worktape. An input head is reversed at will. The worktape is written blank symbols in advance.

Definition 9 (DTM)
A deterministic Turing machine is 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, where

- Q : a finite set of states
- Σ : a finite set of input symbols
- Γ : a finite set of worktape symbols
- q_0 : the start state($q_0 \in Q$)
- B : the blank symbol($B \in \Gamma$)
- F : a set of final states($F \subseteq Q$)
- δ : a transition function $(Q \times \Sigma \times \Gamma) \rightarrow (Q \times \Gamma \times D_i \times D_w)$,
  where D_i is the set of possible input head moves, D_w is the set of possible worktape head moves, $D_i(D_w) = \{-1, 0, 1\}$, where $-1, 1, 0$ denotes head moves Left, Right and No moves respectively.

Suppose both sides of the input tape are enclosed by end markers (\$,#).

The configuration of M denotes C_M and is 4-tuple (q, P_i, P_t, τ), where

- q : a state of the finite control
- P_i : a position of the input head
- P_t : a position of the worktape head
- τ : contents of the worktape

An initial configuration is $C_{initM} = (q_0, 0, 0, BB\ldots B)$, where 0 denotes that the input head and the worktape head are both on left end markers \$. If q is a final state ($q \in F$), the configuration is called an accepted configuration C_{acptM}.

We define a function f which computes the next configuration C'_M from a configuration C_M.

$$f(C_M, x) = C'_M = (q', P_i + d_i, P_w + d_w, \tau'),$$

where x is a symbol under the input head, $d_i, d_w \in \{-1, 0, 1\}$ denotes a direction of a head movement.

C_1 has two next configurations because the input symbol x under the input head of C_1 is 0 or 1. Let us consider next configurations C_2, C_3 of a configuration C_1. Let the state be q, a symbol on the input head $a(\in \Sigma)$, and a symbol on the worktape head $b(\in \Gamma)$ of the configuration C_1. A *transitive function* δ is expressed as follows:

$$\delta(q, a, b) = (q', b', d_i, d_w)$$

Using a function f, we can compute $f(C_1, 0) = C_2$ and $f(C_1, 1) = C_3$ according to the transitive function δ. The symbol $C_2(resp.\ C_3)$ denotes the next configuration of the configuration C_1 when the input symbol of C_1 is 0 (resp. 1) of M. □

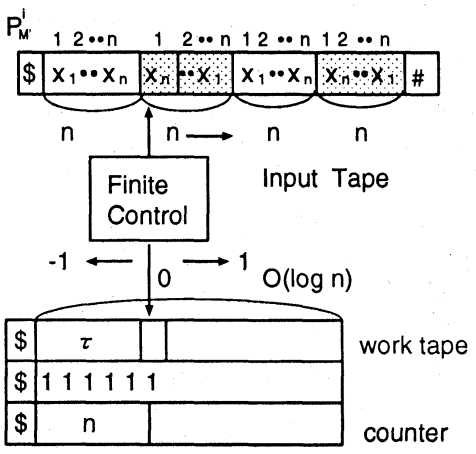

Figure 2: Log-Space Bounded DTM M'

Definition 10 (*DLOG*)
DLOG is the class of languages on $\{0,1\}^*$ that are accepted by logarithmic space bounded DTMs. □

3 Computational Power of PolyI$^{k(n)}$ BDD

In this section, we prove that $PolyI^{k(n)}BDD$ and $O(log\ n)$ space bounded deterministic Turing machines (DTMs) are of the same computational power. We also mention relations between PolyIBDD and formal languages.

3.1 $PolyI^{k(n)}BDD$ and Log-Space Bounded DTM

3.1.1 $DLOG \subseteq PolyIBDD$

Definition 11 (*even turns reverse order expansion* of an input tape)
Let the contents of an input tape be $(b_1,b_2,\ldots,b_n)$. An even turns reverse order expansion of the input tape is such that $(b_1,b_2,\ldots,b_n)$, $(b_n,b_{n-1},\ldots,b_1)$, $(b_1,b_2,\ldots,b_n)$, $(b_n,b_{n-1},\ldots,b_1)$ .... That is, a string of input symbols[4], which has the length n, is written $k(n)$ times repeatedly and even turns of input strings are written in the reversed order.

Lemma 1 For a given $O(log\ n)$ space bounded DTM M with a two-way input tape, there exists an $O(log\ n)$ space bounded DTM M' with a one-way input tape, which is the even turns reverse order expansion of M's input tape, that simulates M. □

Proof.
It is necessary to satisfy the following two conditions in order to prove that M' simulates M.

[4] a string of input symbols is $(b_1,b_2,\ldots,b_n)$ in this case.

1. (Condition 1) Let C^j_M be a j-th step's configuration of M and C^{j+1}_M a $(j+1)$-th step's configuration of M. If $C^j_M \vdash_M C^{j+1}_M$ according to the transitive function δ_M, then there is a succession of transitions $C^j_{M'} \vdash^*_{M'} C^{j+1}_{M'}$. The configuration $C^j_{M'}$ that corresponds to C^j_M is a j step's configuration of M' and $C^{j+1}_{M'}$ that corresponds to C^{j+1}_M is a $(j+1)$-th step's configuration of M'. $(C^j_M \leftrightarrow C^j_{M'}, C^{j+1}_M \leftrightarrow C^{j+1}_{M'})$
2. (Condition 2) There is a configuration graph of M' $(G_{M'})$ which corresponds to G_M of M.

We show M' in Figure 2. M' is a deterministic Turing machine with a one-way read-only input tape and a two-way log-space bounded worktape. Let the contents of the input tape of M be $(b_1,b_2,\ldots,b_n)$ and the reversal of M $k(n)-1$[5] times. M' has the following properties.

- An input tape of M' is written such that $(b_1,b_2,\ldots,b_n)$,$(b_n,b_{n-1},\ldots,b_1)$, $(b_1,b_2,\ldots,b_n)$,$(b_n,b_{n-1},\ldots,b_1)$ .... A string of input symbols, which has the length n, is written $k(n)$ times repeatedly and even turns of input strings are written in the reversed order.
- The input head position of M' denotes $P^i_{M'}$. $P^i_{M'}$ shows a distance from the left end of a input string. $(1 \le P^i_{M'} \le n)$
- The input head of M' moves right, or halts.
- The length of the worktape has $O(log\ n)$. The head of the worktape of M' moves right, left, and halts.
- The worktape of M' has three tracks. Two tracks are used to record the symbols such as the worktape of M and to record the head position. The other track is used to count the number of proper movements of M''s input head.

The numbers $-1, 0, +1$ denote input head movements of M: left, no moves, and right respectively. The positive(or negative) number denotes quantities of right(or left) movements. In order for M' to read the same input symbol as M, the $(j+1)$-th steps of M''s input head should be as follows.

- $2(n - P^i_{M'}(j)) + 2$, (if M's input head reverses)
- $0, +1$, (if M's input head does not reverse)

$P^i_{M'}(j)$ $(1 \le P^i_{M'}(j) \le n)$ is the input head position of M' at j-th step. If the input head of M reverses, the input head of M' moves to the right string of inputs.

$P^i_{M'}(j)$ $(1 \le P^i_{M'}(j) \le n)$ is determined below. We make a counter in the worktape of M'. Whenever M''s input head moves right, the counter is increased one by one. If the counter equals $n+1$, the counter

[5] We select $k(n)-1$ because of the convenience of the discussion below.

is initialized to 1. $P^i_{M'}(j)$ is determined using this counter.

Let the j-th step's configuration of M be $C^j_M = (q, P^i_M(j), P^w_M(j), \tau)$. The $(j+1)$-th step's configuration C^{j+1}_M becomes
$f(C^j_M, x) = C^{j+1}_M = (q', P^i_M(j) + d^i_M(j+1), P^w_M(j) + d^w_M(j+1), \tau')$.
Suppose the corresponding configuration of M' is $C^j_{M'} = (q, P^i_{M'}(j), P^w_{M'}(j), \alpha\tau)$ (α is a counter part), $C^{j+1}_{M'}$ which satisfies the Condition 1 is determined as follows.

- If M's input head is reversed
  - $d^i_{M'}(j+1) = 2(n - P^i_{M'}(j)) + 2$ when $d^i_M(j) = 1$ and $d^i_M(j+1) = -1$, or $d^i_M(j) = -1$ and $d^i_M(j+1) = 1$.
    $C^{j+1}_{M'} = (q', P^i_{M'}(j) + 2(n - P^i_{M'}(j)) + 2, P^w_{M'}(j) + d^w_{M'}(j+1), \alpha'\tau')$
    $P^i_{M'}(j+1)$, the input position of $C^{j+1}_{M'}$, is determined that the input head moves right $2(n - P^i_{M'}(j)) + 2$ times from the position $P^i_{M'}(j)$. If a counter for head movement equals $n+1$, then the counter is initialized to 1. The range of $P^i_{M'}(j+1)$ is $(1 \leq P^i_{M'}(j+1) \leq n)$. The transition from $C^j_{M'}$ to $C^{j+1}_{M'}$ is expressed by a succession of transitions using the reflexive and transitive closure.
    $C^j_{M'} \vdash^*_{M'} C^{j+1}_{M'}$
- If M's input head is not reversed
  - $d^i_{M'}(j+1) = 1$ when $d^i_M(j+1) = 1$
    $C^{j+1}_{M'} = (q', P^i_{M'}(j) + 1, P^w_{M'}(j) + d^w_{M'}(j+1), \alpha\tau')$
    $C^j_{M'} \vdash_{M'} C^{j+1}_{M'}$
  - $d^i_{M'}(j+1) = 0$ when $d^i_M(j+1) = 0$
    $C^{j+1}_{M'} = (q', P^i_{M'}(j) + 0, P^w_{M'}(j) + d^w_{M'}(j+1), \alpha\tau')$
    $C^j_{M'} \vdash_{M'} C^{j+1}_{M'}$
  - $d^i_{M'}(j+1) = 1$ when $d^i_M(j+1) = -1$
    $C^{j+1}_{M'} = (q', P^i_{M'}(j) + 1, P^w_{M'}(j) + d^w_{M'}(j+1), \alpha\tau')$
    $C^j_{M'} \vdash_{M'} C^{j+1}_{M'}$

We can make a configuration graph G_M using the function $f(C_M, x)$ defined in Section 2.2. The initial configuration of M is an initial node of G_M. The graph is a directed graph. Each node is a configuration of M. If a j-th configuration C^j_M moves to a $(j+1)$-th configuration C^{j+1}_M, C^j_M is connected with C^{j+1}_M by an arc whose value is a symbol on the input head. A (non)accepting configuration is a terminal node. The space of the worktape of M is bounded by $O(\log n)$ for input size n, and thus the number of configurations of M is limited by a polynomial of n at most. The number of nodes in the graph is bounded by a polynomial of n because the graph size equals the number of configurations of M.

The configuration graph of M (G_M) is a graph which connects from a initial configuration to a (non)accepting configuration. We replace configuration nodes of M with configuration nodes of M' such that $C_{initM} \mapsto C_{initM'}$, $C_{acptM} \mapsto C_{acptM'}$, and $C_{nacptM} \mapsto C_{nacptM'}$. We replace successions of configurations of M with successions of configuration of M' such that $C^{\forall j}_M \in G_M$, $C^j_M \vdash_M C^{j+1}_M \mapsto C^j_{M'} \vdash^*_{M'} C^{j+1}_{M'}$. The resultant one is a configuration graph of M' ($G_{M'}$). The number of configurations of M' is restricted to polynomial of n because the replacement of successions of configurations is added to polynomial configurations at most. The number of nodes in the graph is polynomial of n at most.

If the input tape of M has length n and the reversal of M is $k(n) - 1$ times, the length of M''s input tape is enough for $k(n) \times n$. There is a succession of configurations of M' ($C_{initM'} \vdash^*_{M'} C_{acptM'}$) that corresponds to a succession of configurations of M ($C_{initM} \vdash^*_M C_{acptM}$) Thus, the Condition 2 is satisfied.

Therefore, M' simulates M. □

In Figure 2, the length of the input tape shown is $4 \times n$, and a string of input symbols is written 4 times. The reversal of input heads of M occurs three. Notice that the reversal of input heads of M occurs $k(n) - 1$ times though the number of string of input symbols is $k(n)$.

Lemma 2 An IBDD of size polynomial to the input size can be constructed from an $O(\log n)$ space bounded DTM M' with a one-way input tape, which is the even turns reverse order expansion of M's input tape. □

Proof.

By virtue of Lemma 1, we can construct a configuration graph of M' ($G_{M'}$) that corresponds to G_M of M and successions of configurations of M' that corresponds to successions of configurations of M.

Suppose that an input tape of M' is the even turns reverse order expansion of the input tape of M. $IG_{M'}$ is a configuration graph and each node of $IG_{M'}$ is the configuration when an input symbol is first read in all of the M''s configurations. Each node of $IG_{M'}$ is expressed by ${}_jX_i$ which is i-th input string $x_i (1 \leq i \leq n)$ in j $(1 \leq j \leq k)$ turns of input strings. There is no loop in $IG_{M'}$.

It is easy to transform a graph $IG_{M'}$ into an IBDD. We replace each node ${}_jX_i$ of $IG_{M'}$ with i-th input string x_i in the j-th layer, 0(or 1) arc with 0(or 1) edge, and an accepting(or nonaccepting) configuration with a constant node 1(or 0). The resultant graph is an IBDD. Since nodes of configurations have a polynomial bound of input size n at most, the size of IBDD is polynomial of n at most. In the above construction, the IBDD has $k(n)$ layers. □

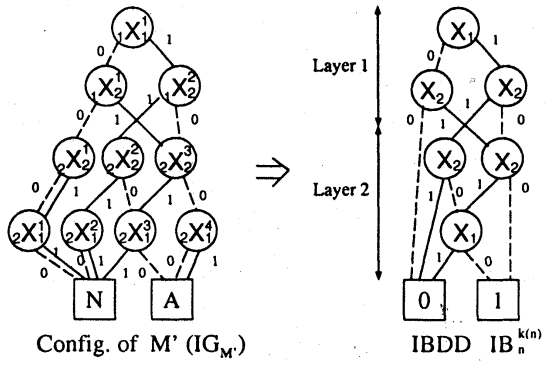

Figure 3: IBDD Construction from a Configuration Graph of M'

We delete nodes v that satisfy $e_0(v) = e_1(v)$ $(v \in N_v)$ in IBDD after we construct an IBDD from a configuration graph of M' ($IG_{M'}$). The Boolean function which is represented by the IBDD is not changed if applicable nodes are all deleted. We show an example that an IBDD constructed from a configuration graph $IG_{M'}$ in Figure 3.

Theorem 1 For a given $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape, there exists a logarithmic uniform sequence of polynomial-sized IBDD with $k(n)$ layers ($PolyI^{k(n)}BDD$) which accepts the language accepted by M, where $k(n)$ is polynomial to the input length n. □

Proof.

Let L be the language that is accepted by $O(log\ n)$ space bounded DTM M. For a given $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape, there is an $O(log\ n)$ space bounded DTM M' with a one-way input tape, which is even turns reverse order expansion of M's input tape, that simulates M by Lemma 1. Thus, L is also accepted by M'. We construct a polynomial-sized IBDD $IB_n^{k(n)}$ from the configuration graph of M' by Lemma 2. An IBDD $IB_n^{k(n)}$ represents a Boolean function $f_{IB_n^{k(n)}}$ and $f_{IB_n^{k(n)}}(b_1, b_2, \ldots, b_n)$ equals 1 if and only if $(b_1, b_2, \ldots, b_n)$ belongs to L. $IB_1^{k(1)}, IB_2^{k(2)}, \ldots, IB_{n-1}^{k(n-1)}, \ldots$ are similarly constructed by Lemma 2. Then we obtain a sequence of IBDDs $\{IB_n^{k(n)}\}$ which accepts L. That is, the sequence of IBDDs $\{IB_n^{k(n)}\}$ is constructed by an $O(log\ n)$ space bounded DTM M' with a one-way input tape which is the even turns reverse order expansion of M's input tape. The sequence is logarithmic uniform by Definition 6. Thus, a logarithmic uniform sequence of polynomial-sized IBDDs ($PolyI^{k(n)}BDD$) of the number of input variables is obtained from $O(log\ n)$ space bounded DTM M. □

The following Corollary 1 and Corollary 2 are derived from Theorem 1.

Corollary 1 $DLOG \subseteq PolyIBDD$ □

Proof.

Let L_i be an arbitrary language which belongs to DLOG. There exists an $O(log\ n)$ space bounded DTM M which accepts L_i. For a given $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape, we can generate $PolyI^{k(n)}BDD$ which accepts the language accepted by M, by Theorem1. The number of layers ($k(n)$) of $PolyI^{k(n)}BDD$ is restricted to $k(n) \leq poly(n)$ at most. $PolyIBDD$ can be expressed as $\bigcup_{k(n)=n^{O(1)}} L_{PolyI^{k(n)}BDD} = PolyIBDD$ by Definition 8. Hence, $PolyIBDD$ includes L_i which belongs to $DLOG$, and thus $DLOG \subseteq PolyIBDD$ is obtained. □

3.1.2 $PolyIBDD \subseteq DLOG$

Theorem 2 For a logarithmic uniform sequence of polynomial size IBDD with $k(n)$ layers ($PolyI^{k(n)}\ BDD$), there exists a $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape which accepts the language accepted by $PolyI^{k(n)}BDD$, where $k(n)$ is polynomial to the input length n. □

Proof.

Let $L_{\{IB_n^{k(n)}\}}$ be an arbitrary language accepted by $PolyI^{k(n)}BDD$. We wish to show that we can construct a logarithmic space bounded DTM M with $k(n) - 1$ times reversal of the input tape which accepts $L_{\{IB_n^{k(n)}\}}$.

Suppose that $\{b_1, b_2, \ldots, b_n\} \in L_{\{IB_n^{k(n)}\}}$ is the contents of the input tape of M. Blank symbols are written on the worktape of M.

First, the input tape scans once to measure its length n and the length is encoded by $O(log\ n)$ bits on the worktape. A node description D_v of the n-th IBDD $IB_n^{k(n)}$ of $PolyI^{k(n)}BDD$ is written on the worktape based on binary encoding of n. Assume that the variable order of odd layers are $(x_1, x_2, \ldots, x_n)$ and that of even layers are $(x_n, x_{n-1}, \ldots, x_1)$ in the IBDD $IB_n^{k(n)}$.

Suppose that a node description $D_v (v \in N_c \cup N_v)$ is written on the worktape. D_v is described a 4-tuple $(index(v), level, e_0(v), e_1(v))$ by Definition 4. A variable x_i $(1 \leq i \leq n)$ of the node v is obtained by $index(v)$. The value $b_{i'}$ $(1 \leq i' \leq n)$ of the variable x_i determines an edge of the node v. If $b_{i'} = 0$ (or $b_{i'} = 1$), then the node v' which enters from $e_0(v)$(or $e_1(v)$) is selected as the next one. If nodes are in odd(or even) layers of an IBDD, M's input head moves right(or left). If $e_0(v)$(or $e_1(v)$) is a terminal node and the $index$ of $e_0(v)$(or $e_1(v)$) is 1, then the input $(b_1, b_2, \ldots, b_n)$ is accepted. If $e_0(v)$(*or* $e_1(v)$) is a terminal node and the $index$ of $e_0(v)$(or $e_1(v)$) is 0,

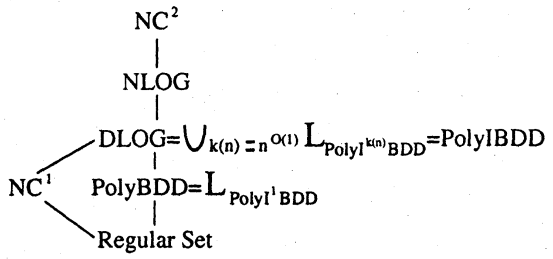

Figure 4: Classes of Formal Languages and PolyIBDD

then the input $(b_1, b_2, \ldots, b_n)$ is not accepted. The logarithmic space bounded DTM M scans the input tape along a path from the root node to a terminal node of IBDD. The input head reverses when two nodes are in different layers. Therefore, we can construct a logarithmic space bounded DTM M which accepts the language accepted by $PolyI^{k(n)}BDD$. □

Corollary 2 is derived from Theorem 2.

Corollary 2 $PolyIBDD \subseteq DLOG$ □

Proof.

Let $L_{\{IB_n^{k(n)}\}}$ be an arbitrary language accepted by $PolyI^{k(n)}BDD$. We wish to show that $L_{\{IB_n^{k(n)}\}}$ belongs to DLOG. By Theorem2, we can construct a $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape which accepts the language accepted by $PolyI^{k(n)}BDD$, where $k(n)$ is a polynomial of the input length n. $PolyIBDD$ is expressed as $\bigcup_{k(n)=n^{O(1)}} L_{PolyI^{k(n)}BDD} = PolyIBDD$ by Definition 8. Therefore, a logarithmic space bounded DTM M with input size n accepts languages that belong to $PolyIBDD$. □

Theorem 3 is derived from Corollary 1 and Corollary 2.

Theorem 3 $PolyIBDD = DLOG$ □

3.2 Classes of Formal Languages and PolyIBDD

We show the proved relations between classes of formal languages and PolyIBDD in Figure 4. PolyBDD[4] is the class of languages that are accepted by logarithmic uniform sequences of polynomial-sized BDD's of input size n. PolyBDD is equivalent to $PolyI^1BDD$.

4 Conclusion

In this paper, we introduced a BDD with redundant variables as an Indexed BDD(IBDD) and defined PolyIBDD as the class of Boolean functions represented by polynomial-sized IBDDs of the number of input variables. We have discussed the relation between PolyIBDD and the class of languages accepted by Turing machines. Results which are obtained in this paper are summarized below.

- For a given $O(log\ n)$ space bounded DTM M with $k(n) - 1$ times reversal of the input tape, there is a uniform sequence of polynomial size IBDD with $k(n)$ layers which accepts the language accepted by M, where $k(n)$ is polynomial to the input length n, and vice versa.
- The following relation also holds.

$$PolyIBDD = DLOG$$

The result mentioned above states that we can construct polynomial-sized IBDD for languages accepted by logarithmic space bounded DTM. The property of PolyBDD, which is the class of Boolean functions which are represented by polynomial-sized BDDs of the number of input variables n, are equal to that of L_{PolyI^1BDD}. That is, $L_{PolyI^{k(n)}BDD}$ is a generalization of PolyBDD.

Acknowledgments

We wish to express our gratitude to the following people: Dr. Nagisa Ishiura of Osaka Univ., Dr. Masahiro Fujita of Fujitsu Lab. of America, Dr. Kiyoharu Hamaguchi of Kyoto Univ., Dr. Hiroyuki Ochi of Kyoto Univ., Yasuhiko Takenaga of Kyoto Univ., Mr. Hiroshi Sawada of NTT, Shin-ichi Minato of NTT, Dr. Kazuaki Murakami of Kyushu Univ. and Dr. Mizuho Iwaihara of Kyushu Univ..

References

[1] Sheldon B. Akers. "Binary Decision Diagrams". *IEEE Trans. Comput.*, Vol. C-27, No. 6, pp. 509–516, Jun. 1978.

[2] Randal E. Bryant. "Graph-Based Algorithms for Boolean Function Manipulation". *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677–691, Aug. 1986.

[3] Randal E. Bryant. "On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication". *IEEE Trans. Comput.*, Vol. 40, No. 2, pp. 205–213, Feb. 1991.

[4] Nagisa Ishiura and Shuzo Yajima. "A Class of Logic Functions Expressible by Polynomial-Size Binary Decision Diagrams". *Technical Report of IPSJ AL20-5*, pp. 1–7, Mar. 1991. (in Japanese).

[5] Jawahar Jain, Magdy Abadir, James Bitner, Donald S. Fussel, Jacob A. Abraham. "IBDDs: An Efficent Functional Representation for Digital Circuits". *Proc. European Design Automation Conference*, pp. 440–446, Mar. 1992.

[6] Satoru Miyano. *"Parallel Algorithms"*. kindai-kagakusya, 1993. (in Japanese).

[7] Hiroto Yasuura. "Theory of the Complexity of Logic Functions and its Application to Logic Design of High-Speed Logic Circuits". *Trans. IPSJ*, Vol. 21, No. 4, pp. 268–278, Jul. 1980. (in Japanese).